REMARKS

Reconsideration of the above-identified patent application in view of the amendments above and the remarks following is respectfully requested.

Claims 19, 22-27, 36 and 37 are in this case. Claims 19, 22-27, 36 and 37 have been rejected under § 103(a). Independent claims 19, 22 and 24-27 have been amended. New independent claim 38 has been added.

The claims before the Examiner are directed toward flash-based units for providing boot code to be executed by external processors and toward related systems and methods. Boot code, including code for basic initialization of system hardware and a command for loading more boot code, is stored in a flash memory that cannot be directly executed. The basic initialization code is transferred to a volatile memory component for execution by a processor to boot the system. The volatile memory component is only large enough to store the basic initialization code.

§ 103(a) Rejections – Gibson et al. '167

The Examiner has rejected claims 19, 22-27, 36 and 37 under § 103(a) as being unpatentable over Gibson et al., US Patent No. 6,601,167 (henceforth, "Gibson et al. '167"). The Examiner's rejection is respectfully traversed.

Gibson et al. '167 teach a motherboard **12**, for a computer system **10**, in which boot code is stored in a sequential access ("UltraNAND") nonvolatile memory **32**. The first page of the boot code is instructions that, when executed by a CPU **14**, load the rest of the boot code from UltraNAND **32** to a RAM **16** for execution. When system **10** is powered up, a boot loader **34** generates setup commands to UltraNAND **32** that instruct UltraNAND **32** to load the first page of the boot code into the output data register of UltraNAND **32**, so that the first page of the boot code is the first code executed by CPU **14**.

The Examiner has cited Gibson et al. '167 column 5 line 65 through column 6 line 12 as teaching that flash memory with on-chip (presumably executable, like RAM **16**) volatile memory is known in the art so that it would be obvious, as an alternative to the invention of Gibson et al. '167, to load the first page of boot code to that volatile memory for execution, thereby allegedly duplicating the present invention. Applicant presumes that the Examiner meant to cite the self-contained paragraph in column 5 line <u>66</u> through column 6 line <u>11</u>. Actually, column 5 line 66 through column 6 line 11 of Gibson et al. '167 says nothing at all about <u>volatile</u> memory. This is what column 5 line 66 through column 6 line 11 of Gibson et al. '167 says:

> Preferably, the state machine and associated functionality of the boot loader **34** can be implemented together with the sequential access memory within a single integrated circuit **40** as indicated at **32'** and **34'** in FIG. **8**. This can be accomplished, for example, by adding sufficient logic circuitry to the sequential access memory to perform the required boot loader function. This embodiment is simpler and requires less device modification than integrating some small portion of <u>random access memory</u> with the sequential memory. This is because the state machine does not require any additional process steps or additional address pins. The UltraNAND or other sequential access memory can remain pin compatible with other memories of similar type. (emphasis added)

It is clear that this "random access memory" is <u>non-volatile</u> random access memory of the kind that Gibson et al. '167 describe in column 1 lines 30-36 as being used in the prior art to store boot code:

> A boot code ROM is typically a random access memory in which an instruction or data at any address can be accessed directly and independently. This supports the branching behavior of most programs in which an instruction following a branch can be read from any arbitrary location in the memory. These memories are also ready to read very shortly after power is supplied.

The solution described by Gibson et al. '167 in column 5 line 66 through column 6 line 11 as inferior to their own solution is to include a small amount of such boot code ROM in UltraNAND **32**, just enough to store the first page of boot code, and to have CPU **14** execute <u>that</u> boot code in place to load the rest of the boot code from UltraNAND **32** to RAM **16** for execution. Such a solution would have only one of the three disadvantages of random access ROM cited in column 1 lines 51-55: the requirement of more physical pins. Only enough boot code ROM to store one page of boot code would be needed, so the high cost and low density of random access ROM relative to sequential access non-volatile memory would be only marginally problematic.

Thus, the present invention, as recited in independent claims 19, 22 and 24-27, is not obvious from Gibson et al. '167.

Nevertheless, in order to expedite the prosecution, Applicant has chosen to amend independent claims 19, 22 and 24-27 to recite another aspect of the present invention that is not obvious from Gibson et al. '167. The <u>only</u> function of the initial boot code of Gibson et al. '167 is to load the rest of the boot code into RAM **16**. By contrast, the initial boot code that is loaded from the nonexecutable flash memory of the present invention to the volatile memory of the present invention initializes the <u>hardware</u> of the system being booted.

Support for this amendment is found in the specification on page 10 lines 15-17:

> S-RAM **40** is optionally very small, such that the copied code is preferably only sufficient for permitting the basic initialization of system **30**...

understood in light of the definition of "initialization" on page 2 lines 2-4:

Typical "boot" operations include initialization of the hardware components of the computational device and also loading of required software program(s).

With independent claims 19 and 22 allowable in their present form it follows that claims 23, 36 and 37 that depend therefrom also are allowable.

In addition, new independent claim 38 has been added. New claim 38 is claim 22 with the additional limitation that the first portion of the boot code is transferred to the volatile memory by a logic separate from the processor. Support for this limitation is found in the specification in Figure 2, that shows logic **42** separate from CPU **32**, and in the description of the booting process on page 10 lines 10-15:

When the "power-on" signal is received, indicating that system **30** should now "boot up", a busy signal on bus **38** signals CPU **32** not to begin operation (stage 1). Next, a specific code...is copied automatically from flash memory **14** to S-RAM **40**, without the intervention of CPU **32** (stage 2). For example, such copying of data could optionally be controlled by logic **42**.

In rejecting claim 1 of US Patent Application Ser. No. 10/888,012, a continuation of the above-identified patent application, the Examiner identified logic **42** with boot loader **34** of Gibson et al. '167. This is an incorrect identification. Boot loader **34** does not load any code. Boot loader **34** merely sends setup commands to UltraNAND **32** that prompt UltraNAND **32** to load code into its own output data register. Then, by executing the code thus loaded into the output data register of UltraNAND **32**, CPU **14** loads the rest of the boot code into RAM **16**. CPU **14** is the only component of motherboard **12** that loads code from UltraNAND **32** to another component of motherboard **12**. There is neither a hint nor a suggestion in Gibson et al. '167 of using boot loader **34** or its equivalent to transfer code from one component of motherboard **12** to another component of motherboard **12**.

In view of the above amendments and remarks it is respectfully submitted that independent claims 19, 22, 24-27 and 38, and hence dependent claims 23, 36 and 37 are in condition for allowance. Prompt notice of allowance is respectfully and earnestly solicited.

Respectfully submitted,

[illegible]

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

Date: April 19, 2007